Checking and Managing Page Cache Usage

When a PBS job is running, the Linux operating system uses part of the physical memory of the compute nodes to store data that is either read from disk or written to disk. The memory used in this way is called *page cache*. In some cases, the amount or distribution of the physical memory used by page cache can affect job performance.

For more information, see the Wikipedia entry on Page cache.

Checking Page Cache Usage

To find out how much physical memory is used by page cache in a node your PBS job is running on, connect to the node via SSH and then run one of the following commands:

- **cat /proc/meminfo**
- **top -b -n1 | head**

Examples

The following example shows output of the **cat /proc/meminfo** command on a node of a running job. The output reports that 60,023,544 KB (~60 GB) of the node's memory is used by page cache:

```
% cat /proc/meminfo
MemTotal:      132007564 kB
MemFree:          605680 kB
Buffers:               0 kB
Cached:        60023544 kB
SwapCached:            0 kB
```

The next example shows output from the **top -b -n1 | head** command, run on the same node as in the previous example. The output reports that the node has 128,913 MB total memory, that 128,314 MB of the total memory is used, and that 59,220 MB (~59 GB) of the used memory is occupied by page cache:

```
% top -b -n1 | head
top - 15:04:00 up 3 days, 16:22,  2 users,  load average: 1.29, 1.23, 1.38
Tasks: 546 total,   2 running, 543 sleeping,   0 stopped,   1 zombie
Cpu(s):  0.5%us,  2.7%sy,  0.0%ni, 96.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    128913M total,   128314M used,      598M free,        0M buffers
Swap:         0M total,        0M used,        0M free,    59220M cached

PID     USER  PR  NI  VIRT  RES  SHR S   %CPU %MEM   TIME+ COMMAND
```

Managing Page Cache

The amount of physical memory used as page cache can be significant if your code reads a large amount of data from disk or writes a large amount of data to disk. For this reason, you may want to manage page cache using one of the methods described in this section.

Assigning Rank 0 to a Single Node

In the case of an MPI application where I/O is done solely by the rank 0 process, it is possible for the page cache to fill up the physical memory of a node, leaving no memory for the other MPI ranks. To avoid this situation, you can assign the rank 0 process to a node by itself. For

example, the following PBS resource request allows for 49 MPI ranks with rank 0 to run on one node, and assigns 24 MPI ranks to each of the other two nodes:

```
#PBS -l select=1:mpiprocs=1:model=has+2:mpiprocs=24:model=has
```

Limiting Page Cache Usage

If there is enough physical memory on the head node to accommodate both the page cache and the memory used by other ranks, keep in mind that the total amount of memory in a node is split between two sockets. For example, the 128 GB of memory in a Pleiades Haswell node is evenly divided between the node's two sockets, as shown in the Haswell configuration diagram. Therefore, if other MPI processes are running on the same socket as the rank 0 process, you should check whether the amount of memory on one socket is enough for both the rank 0 process and the other ranks. If there is not enough memory, the other ranks may be forced to allocate and use memory on the other socket, causing a non-local memory access that may result in degraded performance.

In cases like this, you can set a limit for page cache usage to ensure there will be enough memory to accommodate both the I/O from the rank 0 process and the amount of memory used by other ranks on the same socket. You can accomplish this by running **`pcachem`**, a page cache management tool. For example, to set the limit at 8 GB (2^{33} bytes = 8589934592 bytes) run:

```
module load pagecache-management/0.5
setenv PAGECACHE_MAX_BYTES   8589934592

mpiexec -n XX pcachem -- a.out
```

where *XX* represents the total number of MPI processes. The exact value that you specify for **`PAGECACHE_MAX_BYTES`** is not important, however, be sure you specify a value that leaves enough physical memory on the socket (after subtracting the amount used by page cache) to accommodate the other MPI ranks on the socket.

Note: You must also load an MPI module for running MPI applications.

If you are using OVERFLOW, consider the option shown in the following example, which demonstrates the use of both page cache management (with **`pcachem`**) and process binding (with **`mbind.x`**) for better performance:

```
module load pagecache-management/0.5
setenv PAGECACHE_MAX_BYTES   8589934592
setenv MBIND /u/scicon/tools/bin/mbind.x

$HOME/bin/overrunmpi -v -n XX -aux "pcachem -- $MBIND -v -gm -cs" YY
```

where *YY* refers to the input file for OVERFLOW. The **`mbind.x`** option **`-gm`** prints memory usage for each process, and the **`-cs`** option distributes the MPI processes among the hardware cores.

For more information about **`mbind.x`**, see Using the mbind Tool for Pinning.

Article ID: 505
Last updated: 09 Aug, 2019
Revision: 50
Running Jobs with PBS -> Optimizing/Troubleshooting -> Managing Memory -> Checking and Managing Page Cache Usage
https://www.nas.nasa.gov/hecc/support/kb/entry/505/